Efficient timed diagnosis using automata with timed domains

Patricia Bouyer, Samy Jaziri, Nicolas Markey

Efficient timed diagnosis using automata with timed domains*

Patricia Bouyer[1], Samy Jaziri[1], and Nicolas Markey[2]

[1] LSV – CNRS & Univ. Paris-Saclay – France
[2] IRISA – Univ. Rennes & CNRS & INRIA – France

Abstract. We consider the problems of efficiently diagnosing and predicting what did (or will) happen in a partially-observable one-clock timed automaton. We introduce *timed sets* as a formalism to keep track of the evolution of the reachable configurations over time, and use our previous work on automata over timed domains to build a candidate diagnoser for our timed automaton. We report on our implementation of this approach compared to the approach of [Tripakis, *Fault diagnosis for timed autmata*, 2002].

1 Introduction

Formal methods in verification. Because of the wide range of applications of computer systems, and of their increasing complexity, the use of formal methods for checking their correct behaviours has become essential [16,10]. Numerous approaches have been introduced and extensively studied over the last 40 years, and mature tools now exist and are used in practice. Most of these approaches rely on building mathematical models, such as automata and extensions thereof, in order to represent and reason about the behaviours of those systems; various algorithmic techniques are then applied in order to ensure correctness of those behaviours, such as *model checking* [11,12], *deductive verification* [17,13] or *testing* [25].

Fault diagnosis. The techniques listed above mainly focus on assessing correctness of the set of all behaviours of the system, in an offline manner. This is usually very costly in terms of computation, and sometimes too strong a requirement. *Runtime verification* instead aims at checking properties of a running system [19]. *Fault diagnosis* is a prominent problem in runtime verification: it consists in (deciding the existence and) building a diagnoser, whose role is to monitor real executions of a (partially-observable) system, and decide *online* whether some property holds (e.g., whether some unobservable fault has occurred) [24,26]. A diagnoser can usually be built (for finite-state models) by determinizing a model of the system, using the powerset construction; it will keep track of all possible states that can be reached after each (observable) step of the system, thereby computing whether a fault may or must have occurred.

* Work supported by ERC project EQualIS.

The related problem of *prediction*, a.k.a. prognosis, (that e.g. no fault may occur in the next five steps) [15], is also of particular interest in runtime verification, and can be solved using similar techniques.

Verifying real-time systems. Real-time constraints often play an important role for modelling and specifying correctness of computer systems. Discrete models, such as finite-state automata, are not adequate to model such real-time constraints; timed automata [1], developed at the end of the 1980's, provide a convenient framework for both representing and efficiently reasoning about computer systems subject to real-time constraints. Efficient offline verification techniques for timed automata have been developed and implemented [3,2]. Diagnosis of timed automata however has received less attention; this problem is made difficult by the fact that timed automata can in general not be determinized [27,14]. This has been circumvented by either restricting to classes of determinizable timed automata [6], or by keeping track of all possible configurations of the automaton after a (finite) execution [26]. The latter approach is computationally very expensive, as one step consists in maintaining the set of all configurations that can be reached by following (arbitrarily long) sequences of unobservable transitions; this limits the applicability of the approach.

Our contribution. In this paper, we (try to) make the approach of [26] more efficiently applicable (over the class of one-clock timed automata). Our improvements are based on two ingredients: first, we use *automata over timed domains* [7] as a model for representing the diagnoser. Automata over timed domains can be seen as an extension of timed automata with a (much) richer notion of time and clocks; these automata enjoy determinizability. The second ingredient is the notion of *timed sets*: timed sets are pairs (E, I) where E is any subset of $\mathbb{R}$, and I is an interval with upper bound $+\infty$; such a timed set represents a set of clock valuations evolving over time: the timed set $(E; I)$ after a delay d represents the set $(E + d) \cap I$. As we prove, timed sets can be used to finitely represent the evolution of the set of all reachable configurations after a finite execution.

In the end, our algorithm can compute a finite representation of the reachable configurations after a given execution, as well as all the configurations that can be reached from there after any delay. This can be used to very quickly update the set of current possible configurations (which would be expensive with the approach of [26]). Besides diagnosis, this can also be used to efficiently predict the occurrence of faults occurring after some delay (which is not possible in [26]). We implemented our technique in a prototype tool: as we report at the end of the paper, our approach requires heavy precomputation, but can then efficiently handle delay transitions.

Related works. Model-based diagnosis has been extensively studied in the community of discrete-event systems [23,24,28]. This framework gave birth to a number of ramifications (e.g. active diagnosis [22], fault prediction [15], opacity [18]), and was applied in many different settings besides discrete-event systems (e.g.

Petri nets, distributed systems [4], stochastic systems [20,5], discrete-time systems [9], hybrid systems [21]).

Much fewer papers have focused on continuous-time diagnosis: Tripakis proposed an algorithm for deciding diagnosability [26]. Cassez developed a uniform approach for diagnosability of discrete and continuous time systems, as a reduction to Büchi-automata emptiness [8]. A construction of a diagnoser for timed systems is proposed in [26]: the classical approach of determinizing using the powerset construction does not extend to timed automata, because timed automata cannot in general be determinized [27,14]. Tripakis proposed the construction of a diagnoser as an online algorithm that keeps track of the possible states and zones the system can be in after each event (or after a sufficiently-long delay), which requires heavy computation at each step and is hardly usable in practice. Bouyer, Chevalier and D'Souza studied a restricted setting, only looking for diagnosers under the form of deterministic timed automata with limited resources [6].

2 Definitions

2.1 Intervals

In this paper, we heavily use intervals, and especially unbounded ones. For $r \in \mathbb{R}$, we define

$$\uparrow\!r = [r; +\infty) \qquad \uparrow r = (r; +\infty) \qquad \downarrow r = (-\infty; r) \qquad \downarrow\!r = (-\infty; r].$$

We let $\widehat{\mathbb{R}}_{\geq 0} = \{\uparrow\!r, \uparrow r \mid r \in \mathbb{R}_{\geq 0}\}$ for the set of upward-closed intervals of $\mathbb{R}_{\geq 0}$; in the sequel, elements of $\widehat{\mathbb{R}}_{\geq 0}$ are denoted with $\widehat{r}$. Similarly, we let $\underset{\smile}{\mathbb{R}}_{\geq 0} = \{\downarrow\!r, \downarrow r \mid r \in \mathbb{R}_{\geq 0}\}$, and use notation $\underset{\smile}{r}$ for intervals in $\underset{\smile}{\mathbb{R}}_{\geq 0}$. The elements of $\widehat{\mathbb{R}}_{\geq 0}$ can be (totally) ordered using inclusion: we write $\widehat{r} \prec \widehat{r}'$ whenever $\widehat{r}' \subset \widehat{r}$ (so that $r < r'$ entails $\uparrow r \prec \uparrow r'$).

2.2 Timed automata

Let Σ be a finite alphabet.

Definition 1. *A* one-clock timed automaton *over Σ is a tuple $\mathcal{A} = (S, \{s_0\}, T, F)$, where S is a finite set of states, $s_0 \in S$ is the initial state, $T \subseteq S \times \widehat{\mathbb{R}}_{\geq 0} \times \underset{\smile}{\mathbb{R}}_{\geq 0} \times (\Sigma \uplus \{\epsilon\}) \times \{0, id\} \times S$ is the set of transitions, and $F \subseteq S$ is a set of final states. A configuration of $\mathcal{A}$ is a pair $(s, v) \in S \times \mathbb{R}_{\geq 0}$. There is a transition from (s, v) to (s', v') if*

- *either $s' = s$ and $v' \geq v$. In that case, we write $(s, v) \xrightarrow{d} (s, v')$, with $d = v' - v$, for such* delay transitions *(notice that we have no invariants);*
- *or there is a transition $e = (s, \widehat{l}, \underset{\smile}{u}, a, r, s')$ s.t. $v \in \widehat{l} \cap \underset{\smile}{u}$ and $v' = v$ if $r = id$, and $v' = 0$ otherwise. For those* action transitions*, we write $(s, v) \rightarrow_e (s', v')$. We assume that for each transition $e = (s, \widehat{l}, \underset{\smile}{u}, a, r, s')$, it holds $\widehat{l} \cap \underset{\smile}{u} \neq \emptyset$.*

Fix a one-clock timed automaton $\mathcal{A}$; we write T_{id} for the set of *non-resetting* transitions, i.e., having *id* as their fifth component, and T_0 for the complement set of *resetting* transitions.

For a transition $e = (s, \widehat{l}, \underset{\smile}{u}, a, r, s')$, we write $\widehat{e}$ and $\underset{\smile}{e}$ for $\widehat{l}$ and $\underset{\smile}{u}$, respectively. We write $\mathsf{src}(e) = s$ and $\mathsf{tgt}(e) = s'$, and $\mathsf{lab}(e) = a \in \Sigma$. We extend these definitions to sequences of transitions $w = (e_i)_{0 \leq i < n}$ as $\mathsf{src}(w) = \mathsf{src}(e_0)$, $\mathsf{tgt}(w) = \mathsf{tgt}(e_{n-1})$, and $\mathsf{lab}(w) = (\mathsf{lab}(e_i))_{0 \leq i < n}$.

Let w be a sequence $(e_{2i+1})_{0 \leq 2i+1 < n}$ of transitions of T, and $d \in \mathbb{R}_{\geq 0}$. We write $(s, v) \xrightarrow{d}_w (s', v')$ if there exist finite sequences $(s_i, v_i)_{0 \leq i \leq n} \in (S \times \mathbb{R}_{\geq 0})^{n+1}$ and $(d_{2i})_{0 \leq 2i < n} \in \mathbb{R}_{\geq 0}^{\lfloor n/2 \rfloor}$ such that $\sum_{0 \leq 2i < n} d_{2i} = d$, and $(s_0, v_0) = (s, v)$ and $(s_n, v_n) = (s', v')$, and for all $0 \leq j < n$, $(s_j, v_j) \xrightarrow{d_j} (s_{j+1}, v_{j+1})$ if j is even and $(s_j, v_j) \to_{e_j} (s_{j+1}, v_{j+1})$ if j is odd. We write $(s, v) \to (s', v')$ when $(s, v) \xrightarrow{d}_w (s', v')$ for some $w \in T^*$ and some $d \in \mathbb{R}_{\geq 0}$.

For any $\lambda \in \Sigma^*$ and any $d \in \mathbb{R}_{\geq 0}$, we write $(s, v) \xrightarrow{d}_\lambda (s', v')$ whenever there exists a sequence of transitions w such that $\lambda = \mathsf{lab}(w)$ and $(s, v) \xrightarrow{d}_w (s', v')$. Notice that[3] $(s, v) \xrightarrow{d}_\perp (s', v')$ (sometimes simply written $(s, v) \xrightarrow{d} (s', v')$) indicates a delay-transition (hence it must be $s = s'$). The untimed language $\mathcal{L}(\mathcal{A})$ of $\mathcal{A}$ is the set of words $\lambda \in \Sigma^*$ such that $(s_0, 0) \xrightarrow{d}_\lambda (s', v')$ for some $s' \in F$ and $d \in \mathbb{R}_{\geq 0}$.

We borrow some of the formalism of [7], in order to define a kind of *powerset construction* for timed automata. For a one-clock timed automaton $\mathcal{A} = (S, \{s_0\}, T, F)$ on Σ, we write $\mathbb{M} = (\mathcal{P}(\mathbb{R}_{\geq 0}))^S$ for the set of *markings*, mapping states of $\mathcal{A}$ to sets of valuations for the unique clock of $\mathcal{A}$. For a marking $m \in \mathbb{M}$, we write $\mathsf{supp}(m) = \{s \in S \mid m(s) \neq \emptyset\}$. For any $l \in \Sigma$, we define the function $\mathbf{O}_l \colon \mathbb{M} \to \mathbb{M}$ by letting, for any $m \in \mathbb{M}$ and any $s' \in S$,

$$\mathbf{O}_l(m) \colon s' \mapsto \{v' \in \mathbb{R}_{\geq 0} \mid \exists s \in S.\ \exists v \in m(s).\ (s, v) \to_l (s', v')\}.$$

Similarly, for any $d \in \mathbb{R}_{\geq 0}$, we let

$$\mathbf{O}_d(m) \colon s' \mapsto \{v' \in \mathbb{R}_{\geq 0} \mid \exists s \in S.\ \exists v \in m(s).\ (s, v) \xrightarrow{d} (s', v')\}.$$

Notice that $\mathbf{O}_d$ simply shifts all valuations by d.

Definition 2 ([7]). *The* powerset automaton *of a timed automaton $\mathcal{A} = (S, \{s_0\}, T, F)$ is a tuple $\mathbf{D}\mathcal{A} = (\mathcal{P}(S), \{\{s_0\}\}, \mathbf{P}T, \mathbf{P}F)$, where $\mathcal{P}(S)$ is the set of states, $\{s_0\}$ is the initial state, $\mathbf{P}T = \{(q, m, a, q') \in \mathcal{P}(S) \times \mathbb{M} \times \Sigma \times \mathcal{P}(S) \mid q = \mathsf{supp}(m),\ q' = \mathsf{supp}(\mathbf{O}_a(m))\}$ is the set of transitions, and $\mathbf{P}F = \{E \in \mathcal{P}(S) \mid E \cap F \neq \emptyset\}$ is a set of final states.*

Configurations of $\mathbf{D}\mathcal{A}$ are all pairs $(q, m) \in \mathcal{P}(S) \times \mathbb{M}$ for which $q = \mathsf{supp}(m)$. There is a transition from a configuration (q, m) to a configuration (q', m') labelled with $l \in \Sigma \cup \mathbb{R}_{\geq 0}$ whenever $m' = \mathbf{O}_l(m)$. We extend this definition to

[3] In this paper, we write $\perp$ for the empty word (or empty sequences) over any alphabet.

sequences alternating delay- and action transitions, and write $(q,m) \xrightarrow{d}_w (q',m')$ when there is a path from (q,m) to (q',m') following the transitions of w in d time units. Similarly, we write $(q,m) \xrightarrow{d}_\sigma (q',m')$ if $(q,m) \xrightarrow{d}_w (q',m')$ and $\mathsf{lab}(w) = \sigma$.

Following [7], the automaton $\mathbf{D}\mathcal{A}$ is deterministic, and it simulates $\mathcal{A}$ in the sense that given two marking m and m' and a word σ of Σ^*, we have $(\mathsf{supp}(m), m) \xrightarrow{d}_\sigma (\mathsf{supp}(m'), m')$ if, and only if, $m'(s') = \{v' \in \mathbb{R}_{\geq 0} \mid \exists s \in S.\ \exists v \in m(s).\ (s,v) \xrightarrow{d}_\sigma (s',v')\}$ for all $s' \in S$.

2.3 Timed automata with silent transitions

The work reported in [7] only focuses on the case when there are no silent transitions. In that case, for any $d \in \mathbb{R}_{\geq 0}$, the operation $\mathbf{O}_d$ can be computed easily, since it amounts to adding d to each item of the marking (in other terms, for any marking m, any state $s \in S$, and any $v \in \mathbb{R}_{\geq 0}$ such that $v + d \in \mathbb{R}_{\geq 0}$, we have $v \in m(s)$ if, and only if, $v + d \in \mathbf{O}_d(m)(s)$). This leads to an efficient expression of a (deterministic) powerset automaton simulating $\mathcal{A}$.

However fault diagnosis should deal with timed automata with unobservable transitions (unobservable transitions then correspond to internal transitions). So we now assume that Σ contains a special *silent letter* ϵ, whose occurrence is not *visible*. This requires changing the definition of lab: we now let

$$\begin{aligned} \mathsf{lab}(\bot) &= \bot \\ \mathsf{lab}(w \cdot e) &= \mathsf{lab}(w) && \text{if } \mathsf{lab}(e) = \epsilon \\ \mathsf{lab}(w \cdot e) &= \mathsf{lab}(w) \cdot \mathsf{lab}(e) && \text{if } \mathsf{lab}(e) \neq \epsilon \end{aligned}$$

Notice that $\mathsf{lab}(w) \in (\Sigma \setminus \{\epsilon\})^*$. We may write $\rightarrow_\epsilon$ in place of $\rightarrow_\bot$, to stick to classical notations and make it clear that it allows silent transitions.

In that case, $\mathbf{D}\mathcal{A}$ still is a (deterministic) powerset automaton that simulates $\mathcal{A}$, and hence is still a *diagnoser*, but the function $\mathbf{O}_d$ cannot be computed by just shifting valuations by d. In its raw form, the function $\mathbf{O}_d$ can be obtained by the computation of the set of reachable configurations in a delay d by following silent transitions. This is analogous to the method proposed by Tripakis [26], and turns out to be very costly. However, a diagnoser must be able to simulate all possible actions of the diagnosed automaton quickly enough, so that it can be used at runtime. In this paper, we introduce a new data structure called *timed sets*, which we use to represent markings in the timed powerset automaton; as we explain, using timed sets we can compute $\mathbf{O}_d$ more efficiently (at the expense of more precomputations).

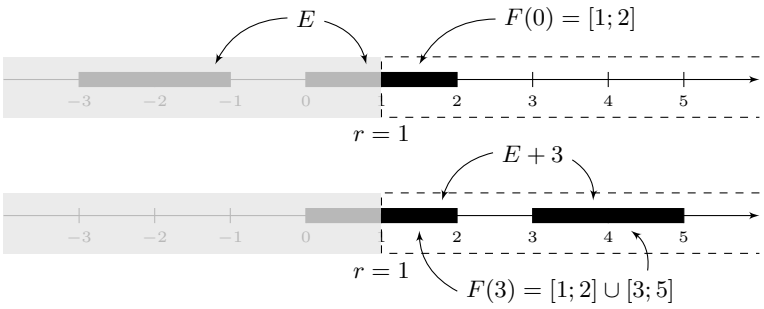

Fig. 1. Example of an atomic timed set $F = ([-3;-1] \cup [0;2]; \uparrow 1)$.

3 (Regular) timed sets

3.1 Timed sets

For a set E and a real d, we define the set $E + d = \{x + d \mid x \in E\}$. We introduce *timed sets* as a way to represent sets of clock valuations (and eventually markings), and their evolution over time.

Definition 3. *An* atomic timed set *is a pair $(E; \widehat{r})$ where $E \subseteq \mathbb{R}$ and $\widehat{r} \in \widehat{\mathbb{R}}_{\geq 0}$. With an atomic timed set $F = (E; \widehat{r})$, we associate a mapping $F \colon \mathbb{R}_{\geq 0} \to 2^{\mathbb{R}}$ defined as $F(d) = (E + d) \cap \widehat{r}$.*

We define the union of two timed sets F and F', denoted as $F \sqcup F'$, as their pointwise union: $(F \sqcup F')(d) = F(d) \cup F'(d)$.

Definition 4. *A* timed set *is a countable set $F = \{F_i \mid i \in I\}$ (intended to be a union, hence sometimes also denoted with $\bigsqcup_{i \in I} F_i$) of atomic timed sets. With such a timed set, we again associate a mapping $F \colon \mathbb{R}_{\geq 0} \to 2^{\mathbb{R}_{\geq 0}}$ defined as $F(d) = \bigcup_{i \in I} F_i(d)$. A timed set is* finite *when it is made of finitely many atomic timed sets. We write $\mathcal{T}(\mathbb{R})$ for the set of timed sets of $\mathbb{R}$.*

Given two timed sets F and F', we write $F \sqsubseteq F'$ whenever $F(d) \subseteq F'(d)$ for all $d \in \mathbb{R}_{\geq 0}$. This is a pre-order relation; it is not anti-symmetric as for instance $(\{1\}; \uparrow 0) \sqsubseteq (\{1\}; \uparrow 1)$ and $(\{1\}; \uparrow 1) \sqsubseteq (\{1\}; \uparrow 0)$. We write $F \equiv F'$ whenever $F \sqsubseteq F'$ and $F' \sqsubseteq F$.

Example 1. Figure 1 displays an example of an atomic timed set $F = (E; \uparrow 1)$, with $E = [-3;-1] \cup [0;2]$. The picture displays the sets $F(0) = [1;2]$ and $F(3) = [1;2] \cup [3;5]$. ◁

3.2 Regular timed sets

In order to effectively store and manipulate timed sets, we need to identify a class of timed sets that is expressive enough but whose timed sets have a finite representation.

Definition 5. *A* regular union of intervals *is a* 4*-tuple* $E = (I, J, p, q)$ *where* I *and* J *are finite unions of intervals of* $\mathbb{R}$ *with rational (or infinite) bounds,* $p \in \mathbb{Q}_{\geq 0}$ *is the period, and* $q \in \mathbb{N}$ *is the offset. It is required that* $J \subseteq (-p; 0]$ *and* $I \subseteq \Uparrow(-q \cdot p)$.

The regular union of intervals $E = (I, J, p, q)$ *represents the set (which we still write* E*)* $I \cup \bigcup_{k=q}^{+\infty} J - k \cdot p$.

Regular unions of intervals enjoy the following properties:

Proposition 6. *Let* E *and* E' *be regular unions of intervals,* K *be an interval, and* $d \in \mathbb{Q}$. *Then* $E \cup E'$, $\overline{E}$, $E + d$ *and* $E - K$ *are regular unions of intervals.*

Definition 7. *A* regular timed set *is a finite timed set* $F = \{(E_i; \widehat{r}_i) \mid i \in I\}$ *such that for all* $i \in I$, *the set* E_i *is a regular union of intervals.*

4 Computing the powerset automaton

In this section, we fix a one-clock timed automaton $\mathcal{A} = (S, \{s_0\}, T, F)$ over alphabet Σ, assumed to contain a silent letter ϵ. We assume that some silent transitions are *faulty*, and we want to detect the occurrence of such faulty transitions based on the sequence of actions we can observe. Following [26], this can be reduced to a *state-estimation problem*, even if it means duplicating some states of the model in order to keep track of the occurrence of a faulty transition. In the end, we aim at computing (a finite representation of) the powerset automaton $\mathbf{D}\mathcal{A}$, which amounts to computing the transition functions $\mathbf{O}_l$ for any $l \in \Sigma$ and $\mathbf{O}_d$ for any $d \in \mathbb{R}_{\geq 0}$. Computing $\mathbf{O}_l(m)$ for $l \in \Sigma$ is not very involved: for each state $s \in S$ and each transition e labelled with l with source s and target s', it suffices to intersect $m(s)$ with the guard $\widehat{e} \cap \underset{\smile}{e}$, and add the resulting interval (or the singleton $\{0\}$ if e is a resetting transition) in $\mathbf{O}_l(m)(s')$.

From now on, we only focus on computing $\mathbf{O}_d$, for $d \in \mathbb{R}_{\geq 0}$. For this, it is sufficient to only consider silent transitions of $\mathcal{A}$: we let $U = U_0 \uplus U_{id}$ be the subset of T containing all transitions labelled ϵ, partitioned into those transitions that reset clock x (in U_0), and those that do not (in U_{id}). We write $\mathcal{A}_\epsilon$ for the restriction of $\mathcal{A}$ to silent transitions, and only consider that automaton in the sequel. All transitions are silent in $\mathcal{A}_\epsilon$, but for convenience, we assume that transitions are labelled with their name in $\mathcal{A}$, so that the *untimed language* of $\mathcal{A}_\epsilon$ is the set of sequences of consecutive (silent) transitions firable from the initial configuration.

4.1 Linear timed markings and their ϵ-closure

We use markings to represent sets of configurations; in order to compute $\mathbf{O}_d$, we need to represent the evolution of markings over time. For this, we introduce *timed markings*. A timed marking is a mapping $M \colon S \to (\mathbb{R}_{\geq 0} \to 2^{\mathbb{R}_{\geq 0}})$. For any $s \in S$ and any $d \in \mathbb{R}_{\geq 0}$, $M(s)(d)$ is intended to represent all clock valuations that can be obtained in s after a delay of d time units. For any delay $d \in \mathbb{R}$,

we may (abusively) write $M(d)$ for the marking represented by M after delay d (so that for any $s \in S$ and any $d \in \mathbb{R}_{\geq 0}$, both notations $M(d)(s)$ and $M(s)(d)$ represent the same subset of $\mathbb{R}$).

A special case of timed marking are those timed markings that can be defined using timed sets; timed markings of this kind will be called *linear timed markings* in the sequel. As we prove below, linear timed markings are expressive enough to represent how markings evolve over time in one-clock timed automata. Atomic (resp. finite, regular) timed markings are linear timed markings whose values are atomic (resp. finite, regular) timed sets (we may omit to mention linearity in these cases to alleviate notations). Union, inclusion and equivalence of (timed) markings are defined statewise.

With any marking m, we associate a linear timed marking, which we write $\overrightarrow{m}$ (or sometimes m if no ambiguity arises), defined as $\overrightarrow{m}(s)(d) = \{v+d \mid v \in m(s)\}$. This timed marking is linear since it can be defined e.g. as $\overrightarrow{m}(s) = (m(s); \uparrow 0)$. This timed marking can be used to represent all clock valuations that can be reached from marking m after any delay $d \in \mathbb{R}_{\geq 0}$.

Given a marking m, a delay d and a sequence $w \in U^*$ of silent transitions of $\mathcal{A}_\epsilon$, we define the marking $m \oplus_w d$ as follows:

$$m \oplus_w d \colon s' \mapsto \{v \in \mathbb{R}_{\geq 0} \mid \exists s \in S.\ \exists v \in m(s).\ (s, v) \xrightarrow{d}_w (s', v')\}$$

(remember that w here represents a sequence of silent transitions). This corresponds to all configurations reachable along w from a configuration in m with a delay of exactly d time units. By definition of the transition relation $\rightarrow_w$, for $m \oplus_w d$ to be non-empty, w must be a sequence of consecutive transitions. With this definition, for any sequence w of silent transitions and any marking m, we can define a timed marking $m^w \colon d \mapsto m \oplus_w d$. In particular, for the empty sequence $\bot$, the timed marking $m^\bot$ is equivalent to the timed marking $\overrightarrow{m}$ (hence it is linear).

For any $d \in \mathbb{R}_{\geq 0}$, we define $m \oplus_\epsilon d$ as $(m \oplus_\epsilon d)(s) = \bigcup_{w \in U^*} (m \oplus_w d)(s)$. The marking $m \oplus_\epsilon d$ represents the set of configurations that can be reached after a delay d through sequences of silent transitions. This gives rise to a timed marking, which we write m^ϵ. By definition of $\mathbf{O}_d$, we have $\mathbf{O}_d(m) = m^\epsilon(d)$ for any marking m and any delay d.

The definition is extended to timed markings as follows: for a timed marking M and a delay d, we let

$$M \oplus_w d \colon s' \mapsto \{v' \in \mathbb{R}_{\geq 0} \mid \exists s \in S.\ \exists d_0 \leq d.\ \exists v \in M(d_0)(s).\ (s, v) \xrightarrow{d-d_0}_w (s', v')\},$$

Again, this gives rise to a timed marking $M^w \colon d \mapsto M \oplus_w d$. Observe that for any *linear* timed marking, we have $M^\bot \equiv M$. Notice also that for any marking m, it holds $(\overrightarrow{m})^w \equiv m^w$. We let $M \oplus_\epsilon d$ be the union of all $M \oplus_w d$ when w ranges over U^*, and M^ϵ be the associated timed marking.

Definition 8. *Let M be a timed marking. A timed marking N is an* ϵ-closure *of M if $N \equiv M^\epsilon$. The timed marking M is said* ϵ-closed *if it is an ϵ-closure of itself.*

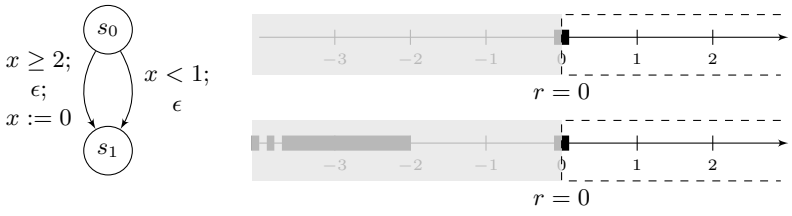

Fig. 2. A silent timed automaton

Our aim in this section is to compute (a finite representation of) an ϵ-closure of any given initial marking (defined using regular unions of intervals).

Example 2. Consider the (silent) timed automaton of Fig. 2. The initial configuration can be represented by the timed marking M with $M(s_0) = ([0;0];\uparrow 0)$ and $M(s_1) = (\emptyset;\uparrow 0)$, corresponding to the single configuration $\{(s_0, x = 0)\}$. This timed marking is not closed under delay- and silent-transitions, as for instance configuration $(s_1, x = 0)$ is reachable; however, this configuration cannot be reached after any delay: it is only reachable after delay 0, or after a delay larger than or equal to 2 time units. In the end, an *ϵ-closed* timed marking for this automaton is $M^\epsilon(s_0) = M(s_0)$, and $M^\epsilon(s_1) = ((-\infty;-2] \cup [0;0];\uparrow 0)$. ◁

4.2 Computing ϵ-closures

Let E and F be two subsets of $\mathbb{R}$. We define their *gauge* as the set $E \bowtie F = (E - F) \cap \mathbb{R}_{\leq 0}$. Equivalently, $E \bowtie F = \{d \in \mathbb{R}_{\leq 0} \mid F + d \cap E \neq \emptyset\}$.

Lemma 9.
- *If $E \leq F$ (in particular, if $E \subseteq \mathbb{R}_{\leq 0}$ and $F \subseteq \mathbb{R}_{\geq 0}$), then $E \bowtie F = E - F$.*
- *if $E > F$, then $E \bowtie F = \emptyset$.*
- *If E and F are two intervals, then $E \bowtie F$ is an interval.*
- *IF E is a regular union of intervals and J is an interval, then $E \bowtie F$ is a regular union of intervals.*
- *If $F' \subseteq \mathbb{R}_{\geq 0}$, then $(E \bowtie F) \bowtie F' = (E \bowtie F) - F'$.*

We now define a mapping $\epsilon \colon \mathcal{T}(\mathbb{R}) \times U^* \to \mathcal{T}(\mathbb{R})$, intended to represent the timed set that is reached by performing sequences of silent transitions from some given timed set. We first consider atomic timed sets, and the application of a single silent transition. The definition is based on the type of the transition:

$$\epsilon((E,\widehat{r}),(s,\widehat{e},\underbrace{e}_{\sim},\epsilon,\mathsf{op},s')) = \begin{cases} (\emptyset;\uparrow 0) & \text{if } \widehat{r} \cap \underbrace{e}_{\sim} = \emptyset \\ (E \cap \underbrace{e}_{\sim};\widehat{p}) & \text{if } \widehat{r} \cap \underbrace{e}_{\sim} \neq \emptyset \text{ and } \mathsf{op} = id \\ (E \bowtie (\widehat{p} \cap \underbrace{e}_{\sim});\uparrow 0) & \text{if } \widehat{r} \cap \underbrace{e}_{\sim} \neq \emptyset \text{ and } \mathsf{op} = 0 \end{cases}$$

where $\widehat{p} = \widehat{r} \cap \widehat{e}$. We extend ϵ to sequences of transitions inductively by letting $\epsilon((E;\widehat{r}),\bot) = (E;\widehat{r})$ and, for $w \in U^+$,

$$\epsilon((E;\widehat{r}), w \cdot e) = \begin{cases} \epsilon(\epsilon((E;\widehat{r}), w), e) & \text{if } \mathsf{tgt}(w) = \mathsf{src}(e) \\ (\emptyset; \uparrow\!0) & \text{otherwise.} \end{cases}$$

Finally, we extend this definition to unions of atomic timed sets by letting $\epsilon(F_1 \sqcup F_2, w) = \epsilon(F_1, w) \sqcup \epsilon(F_2, w)$. We now prove that this indeed corresponds to applying silent- and delay-transitions from a given timed set.

Lemma 10. *Let F be a timed set and $w \in U^*$. Then for any $d \in \mathbb{R}_{\geq 0}$ and any $v \in \mathbb{R}_{\geq 0}$,*

$$v \in (\epsilon(F,w))(d) \;\Leftrightarrow\; \exists d_0 \in [0;d].\ \exists v' \in F(d_0).\ (\mathit{src}(w), v') \xrightarrow{d-d_0}_w (\mathit{tgt}(w), v).$$

Proof. We carry the proof for the case where F is an atomic timed set. The extension to unions of atomic timed sets is straightforward. The proof for atomic timed sets is in two parts: we begin with proving the result for a single transition (the case where $w = \bot$ is easy), and then proceed by induction to prove the full result.

We begin with the case where w is a single transition $e = (s, \widehat{e}, \underset{\sim}{e}, \epsilon, \mathsf{op}, s')$. In case F is empty, then also $\epsilon(F,e)$ is empty, and the result holds. We now assume that F is not empty, and consider three cases:

- if $\widehat{r} \cap \underset{\sim}{e} = \emptyset$, then $\epsilon(F,e) = (\emptyset; \uparrow\!0)$. On the other hand, for any d_0 and any $v' \in F(d_0)$, it holds $v' \in \widehat{r}$, so that $v' \notin \underset{\sim}{e}$, and the transition cannot be taken from that valuation. Hence both sides of the equivalence evaluate to false, and the equivalence holds.
- now assume that $\widehat{r} \cap \underset{\sim}{e} \neq \emptyset$, and consider the case where e does not reset the clock. Then $v \in (\epsilon(F,w))(d)$ means that $v \in (E+d) \cap (\underset{\sim}{e} + d) \cap \widehat{p}$. If such a v exists, then $\widehat{p} \cap \underset{\sim}{e} \cap [v-d; v]$ is non-empty: indeed, this is trivial if either $v \in \underset{\sim}{e}$, or $v - d \in \widehat{p}$, or $d = 0$; otherwise, we have $\underset{\sim}{e} \subseteq \downarrow v$ and $\widehat{p} \subseteq \uparrow v - d$, so that $\widehat{p} \cap \underset{\sim}{e} \subseteq (v-d; v)$. Moreover, $\widehat{p} \cap \underset{\sim}{e} \neq \emptyset$ since $\widehat{r} \cap \underset{\sim}{e} \neq \emptyset$ and $\widehat{e} \cap \underset{\sim}{e} \neq \emptyset$. Then for any v' in that set $\widehat{p} \cap \underset{\sim}{e} \cap [v-d; v]$, letting $d_0 = v' - (v-d)$, we have $v' \in E + d_0$. In the end, $v' \in F(d_0)$, and $v' \in \widehat{e} \cap \underset{\sim}{e}$, so that $(\mathsf{src}(e), v') \xrightarrow{d-d_0}_e (\mathsf{tgt}(e), v)$. Conversely, if $d_0 \in [0;d]$ and $v' \in F(d_0)$ exist such that $(\mathsf{src}(w), v') \xrightarrow{d-d_0}_w (\mathsf{tgt}(w), v)$, then $v' \in E + d_0 \cap \widehat{r}$, and for some $d_1 \leq d - d_0$, $v' + d_1 \in \widehat{e} \cap \underset{\sim}{e}$. Then letting $v = v' + d - d_0$, we have $v \in E + d$ and $v \in \widehat{r} + (d - d_0) \subseteq \widehat{r}$ and $v \in \widehat{e} + (d - d_0 - d_1) \subseteq \widehat{e}$ and $v \in \underset{\sim}{e} + (d - d_0 - d1) \subseteq \underset{\sim}{e} + d$. This proves our result for this case.
- we finally consider the case where $\widehat{r} \cap \underset{\sim}{e} \neq \emptyset$ and e resets the clock. In this case, $v \in (\epsilon(F,w))(d)$ means that $v \in \uparrow\!0$ and $v - d \in E \bowtie (\widehat{p} \cap \underset{\sim}{e})$, which rewrites as $0 \leq v \leq d$ and $(E + d - v) \cap (\widehat{p} \cap \underset{\sim}{e}) \neq \emptyset$. Let $d_0 = d - v$. The property above entails that $0 \leq d_0 \leq d$, and that there exists some $v' \in (E + d_0) \cap (\widehat{p} \cap \underset{\sim}{e})$, so that $0 \leq d_0 \leq d$, $v' \in F(d_0)$ and $(\mathsf{src}(e), v') \xrightarrow{d-d_0}_e (\mathsf{tgt}(e), v)$. Conversely, if those conditions hold, then for some $0 \leq d_1 \leq d - d_0$, we have $v' + d_1 \in \widehat{e} \cap \underset{\sim}{e}$,

and $v = d - (d_0 + d_1)$ (remember that e resets the clock). Then $v' + d_1 \in E + (d_0 + d_1) \cap \widehat{r} \cap \widehat{e} \cap \underbrace{e}$, so that $-(d_0 + d_1) \in E \bowtie (\widehat{p} \cap \underbrace{e})$, and finally $v \in \epsilon(F, w)(d)$.

We now extend this result to sequences of transitions. The case where $w = \bot$ is straightforward. Now assume that the result holds for some word w, and consider a word $w \cdot e$. In case $\mathsf{tgt}(w) \neq \mathsf{src}(e)$, the result is trivial.

The case of single transitions has been handled just above. We thus consider the case of $w \cdot e$ with $w \in U^+$. First assume that $v' \in (\epsilon(F, w \cdot e))(d)$, and let $F' = \epsilon(F, w)$. Then $v' \in (\epsilon(F', e))(d)$, thus there exist $0 \leq d_0 \leq d$ and $v \in F'(d_0)$ s.t. $(\mathsf{src}(e), v) \xrightarrow{d-d_0}_e (\mathsf{tgt}(e), v')$. Since $v \in F'(d_0)$, there must exist $0 \leq d_1 \leq d_0$ and $v'' \in F(d_1)$ such that $(\mathsf{src}(w), v'') \xrightarrow{d_0-d_1}_w (\mathsf{tgt}(w), v)$. We thus have found $0 \leq d_1 \leq d$ such that $(\mathsf{src}(w), v'') \xrightarrow{d-d_1}_{w \cdot e} (\mathsf{tgt}(e), v')$.

Conversely, if $(\mathsf{src}(w), v'') \xrightarrow{d-d_1}_{w \cdot e} (\mathsf{tgt}(e), v')$ for some $0 \leq d_1 \leq d$ and $v'' \in F(d_1)$, then we have $(\mathsf{src}(w), v'') \xrightarrow{d_0-d_1}_w (\mathsf{tgt}(w), v) \xrightarrow{d-d_0}_e (\mathsf{tgt}(e), v')$ for some $d_0 \in [d_1; d]$ and some v. We prove that $v \in (\epsilon(F, w))(d_0)$: indeed, we have $d_1 \in [0; d_0]$, and $v'' \in F(d_1)$ such that $(\mathsf{src}(w), v'') \xrightarrow{d_0-d_1}_w (\mathsf{tgt}(w), v)$, which by induction hypothesis entails $v \in (\epsilon(F, w))(d_0)$. Thus we have $d_0 \in [0; d]$ and $v \in F'(d_0)$, where $F' = \epsilon(F, w)$, such that $(\mathsf{tgt}(w), v) \xrightarrow{d-d_0}_e (\mathsf{tgt}(e), v')$, which means $v' \in \epsilon(F', e)(d)$, and concludes the proof. □

Thanks to this semantic characterization of $\epsilon(F, w)$, we get:

Corollary 11. *For any sequence w of transitions, and any two equivalent timed sets F and F', the timed sets $\epsilon(F, w)$ and $\epsilon(F', w)$ are equivalent.*

Finally, we extend ϵ to linear timed markings in the expected way: given a linear timed marking M, and a sequence w of transitions, we let

$$\epsilon(M, w)\colon \begin{array}{ll} s \mapsto \epsilon(M(\mathsf{src}(w)), w) & \text{if } s = \mathsf{tgt}(w), \\ s \mapsto (\emptyset; \uparrow 0) & \text{otherwise.} \end{array}$$

Again, we have $\epsilon(M_1 \sqcup M_2, w) \equiv \epsilon(M_1, w) \sqcup \epsilon(M_2, w)$ for any $w \in U^*$. Then:

Lemma 12. *$\epsilon(M, w) \equiv M^w$ for all $w \in U^*$ and all linear timed marking M.*

Letting $\epsilon(M, \mathcal{L}) = \bigsqcup_{w \in \mathcal{L}} \epsilon(M, w)$ for any subset $\mathcal{L}$ of U^*, and $\epsilon(M) = \epsilon(M, U^*)$, we immediately get:

Theorem 13. *For any linear timed marking M, it holds $\epsilon(M) \equiv M^\epsilon$.*

It follows that the closure of any marking can be represented as a linear timed marking. However, this linear timed marking is currently defined as an infinite union over all sequences of consecutive silent transitions. We make the computation more effective in the next section.

4.3 Finite representation of the closure

In this section, we prove that we can effectively compute a finite representation of the closure of any regular timed marking. More precisely, we show how to compute such a closure as a regular timed marking.

Finiteness. Indeed, let M be (a finite representation of) a regular timed marking: then M can be written as the finite union of atomic regular timed markings $M_{(s,E,\widehat{r})}$, defined as $M_{(s,E,\widehat{r})}(s) = (E;\widehat{r})$ and $M_{(s,E,\widehat{r})}(s') = (\emptyset;\uparrow 0)$ for all $s' \neq s$. In the end, any regular timed marking M can be written as the finite union $\bigsqcup_{i\in I} M_i$ of atomic regular timed markings. Thus we have $\epsilon(M) \equiv \bigsqcup_{i\in I} \epsilon(M_i)$, and it suffices to compute ϵ for atomic regular timed markings $M_{(s,E,\widehat{r})}$. We prove that those closures can be represented as regular timed markings.

We write $\epsilon_1((E;\widehat{r}),w)$ and $\epsilon_2((E;\widehat{r}),w)$ for the first and second components of $\epsilon((E;\widehat{r}),w)$. Notice that $\epsilon_2((E;\widehat{r}),w)$ does not depend on E (so that we may denote it with $\epsilon_2(\widehat{r},w)$ in the sequel). In particular,

- $\epsilon_2((E;\widehat{r}),w) = \uparrow 0$ if $w \in U^* \times U_0$ is a sequence of consecutive transitions ending with a resetting transition;
- $\epsilon_2((E;\widehat{r}),w) = \widehat{r} \cap \bigcap_{i<k} \widehat{e_i}$ if $w = e_1 \dots e_k \in {U_{id}}^*$ is a sequence of consecutive non-resetting transitions.

Letting $\mathbb{J}_{\widehat{r}} = \{\uparrow 0, \widehat{r}\} \cup \{\widehat{e} \mid e \in U_{id}\}$, it follows that $\epsilon_2((E;\widehat{r}),w) \in \mathbb{J}_{\widehat{r}}$. for any $(E;\widehat{r})$ and any w. Thus $\epsilon(M)$ can be written as a finite union of atomic timed markings.

Regularity. To prove regularity, we first introduce some more formalism:

- we let $\widehat{G_{id}} = \{\widehat{e} \mid e \in U_{id}\}$ and $\underbrace{G_{id}} = \{\underbrace{e} \mid e \in U_{id}\}$. We thus have $\mathbb{J}_{\widehat{r}} = \{\uparrow 0, \widehat{r}\} \cup \widehat{G_{id}}$;
- for $\widehat{r} \in \widehat{\mathbb{R}}_{\geq 0}$ and $e \in U$, we write $\Phi(\widehat{r},e)$ for the interval $\widehat{r} \cap \widehat{e} \cap \underbrace{e}$;
- we define a mapping $\mathcal{J}_{\widehat{r}} \colon U^* \to \mathbb{N}^{\mathbb{J}_{\widehat{r}} \times U_0}$ that counts the number of occurrences of certain timing constraints at resetting transitions along a path: precisely, it is defined inductively as follows (where $\Cup$ represents addition of an element to a multiset):

$$\begin{aligned} \mathcal{J}_{\widehat{r}}(\bot) &= \{0\}^{\mathbb{J}_{\widehat{r}} \times U_0} \\ \mathcal{J}_{\widehat{r}}(w \cdot e) &= \mathcal{J}_{\widehat{r}}(w) \Cup \{(\epsilon_2(\widehat{r},w), e)\} && \text{if } e \in U_0 \\ \mathcal{J}_{\widehat{r}}(w \cdot e) &= \mathcal{J}_{\widehat{r}}(w) && \text{if } e \in U_{id}. \end{aligned}$$

By induction on w, we prove:

Lemma 14. *Let $(E;\widehat{r})$ be a timed set with $E \subseteq \mathbb{R}_{\leq 0}$, and $w \in U^*$. Then*

$$\epsilon_1((E;\widehat{r}),w) = E - \sum_{J=(\widehat{g},e)\in\mathbb{J}_{\widehat{r}}} \mathcal{J}_{\widehat{r}}(w)(J) \times \Phi(\widehat{g},e) \subseteq \mathbb{R}_{\leq 0}.$$

Now, we fix an atomic regular timed marking $M_{(s,E,\widehat{r})}$. For any state s' of $\mathcal{A}_\epsilon$, we let $\mathcal{L}(s,s')$ be the set of all sequences of consecutive transitions from s to s' in $\mathcal{A}_\epsilon$. Then

$$(M_{(s,E,\widehat{r})})^\epsilon = \bigsqcup_{s'\in S} \epsilon(M_{(s,E,\widehat{r})}, \mathcal{L}(s,s')).$$

Hence we need to prove that $\epsilon_1(M_{(s,E,\widehat{r})}, \mathcal{L}(s,s'))$ is regular.

For any set $\mathcal{L}$ of sequences of consecutive transitions, and for any $\widehat{r}$ and $\widehat{r}'$ in $\widehat{\mathbb{R}}_{\geq 0}$, we let $\mathcal{L}_{\widehat{r}}^{\widehat{r}'} = \{w \in \mathcal{L} \mid \widehat{r}' = \epsilon_2(\widehat{r}, w)\}$. One easily observes that for any $\widehat{r} \in \widehat{\mathbb{R}}_{\geq 0}$ and any $\mathcal{L}$, it holds $\mathcal{L} = \bigcup_{\widehat{r}'\in\mathbb{J}_{\widehat{r}}} \mathcal{L}_{\widehat{r}}^{\widehat{r}'}$, so that

$$\epsilon_1(M_{(s,E,\widehat{r})}, \mathcal{L}(s,s')) = \bigcup_{\widehat{r}'\in\mathbb{J}_{\widehat{r}}} \epsilon_1(M_{(s,E,\widehat{r})}, [\mathcal{L}(s,s')]_{\widehat{r}}^{\widehat{r}'}).$$

The following property entails that this set is a regular union of intervals:

Lemma 15. *Let E be a regular union of intervals, $\widehat{r}$ and $\widehat{r}'$ be two elements of $\widehat{\mathbb{R}}_{\geq 0}$, and $\mathcal{L} \subseteq \mathcal{L}(\mathcal{A}_\epsilon)$ be a* regular *language. Then $\epsilon_1(M_{(s,E,\widehat{r})}, \mathcal{L}_{\widehat{r}}^{\widehat{r}'})$ is a regular union of intervals.*

5 Experimentations

In order to evaluate the possible improvement of our approach compared to the diagnoser proposed in [26], we implemented and compared the performances of both approaches. Sources can be downloaded at `http://www.lsv.fr/~jaziri/DOTA.zip`

5.1 Comparison of the approaches

In the approach of [26], the set of possible current configurations is stored as a marking. If an action l occurs after some delay d, the diagnoser computes the set of all possible configurations reached after delay d (possibly following silent transitions), and applies from the resulting markings the set of all available transitions labelled l. This amounts to computing the functions $\mathbf{O}_d$ and $\mathbf{O}_l$ at each observation. There is also a timeout, which makes the diagnoser update the marking (with $\mathbf{O}_d$) regularly if no action is observed. The computation of $\mathbf{O}_d$ is heavily used, and has to be performed very efficiently so that the diagnoser can be used at runtime.

In our approach, we use *timed markings* to store sets of possible configurations. Given a timed marking, when an action l is observed after some delay d, we can easily compute the set of configurations reachable after delay d, and have to apply $\mathbf{O}_l$ and recompute the ϵ-closure. Following [7], $\mathbf{O}_l$ can be performed as a series of set operations on intervals. The ϵ-closure can be performed as a series of subtractions between an interval and regular unions of intervals (see Lemma 14). Those regular unions can be precomputed; while this may require exponential time and space to compute and store, this makes the simulation of a delay transition very efficient.

5.2 Implementation

In our experimentations, in order to only evaluate the benefits of the precomputation and of the use of ϵ-closures in our approach compared to that of [26], we use the same data structure for both diagnosers. In particular, both diagnosers are implemented as automata over timed domains [7], where the timed domain is the set of timed markings. The only difference lies in the functions computing the action- and the delay transitions. As a consequence, both implementations benefit from the data structure we chose for representing timed intervals, which allows us to compute basic operations in linear time. Also, both structures use the same reachability graphs for either computing the sets of reachable configurations or the Parikh images.

Our implementation is written in Python3. One-clock timed automata and both diagnosers are instances of an abstract class of automata over timed markings; timed markings are implemented in a library `TILib`. Simulations of those automata are performed using an object called `ATDRunner`, which takes an automaton over timed markings and simulates its transitions according to the actions it observes on a given *input channel*. It may also write what it does on an *output channel*. A channel is basically a way of communicating with `ATDRunner`s.

In order to diagnose a given one-clock timed automaton, stored in an object `OTAutomata`, we first generate a diagnoser object, either a `DiagOTA` or a `TripakisDOTA`, depending of which version we want to use. Then we launch two threads: one is an `ATDRunner` simulating the timed automaton, listening to some channel object `Input`, and writing every non-silent action it performs on some other channel object `Comm`. The other one is another `ATDRunner` simulating the diagnoser and listening to the `Comm` channel.

In a `DiagOTA` object, which corresponds to our approach, we have already precomputed the relevant timed intervals; action transitions are then made by operations over timed markings, and delay transitions are encoded by increasing a *padding information* on the timed markings, which is applied when performing the next action transition. In such a simulation, we can thus keep track of which states may have been reached, but also predict which states may be reached in the future and the exact time before we can reach them.

In a `TripakisDOTA` object, which corresponds to the approach of [26], action and delay transitions are simulated by computing all configurations reachable through that action or delay, also allowing arbitrarily many silent transitions. This does not allow for prediction.

5.3 Results

Table 3 reports on the performances of both implementations on a small set of (randomly generated) examples. Those examples are given in Appendix B, and are distributed with our prototype. In Table 3, we give the important characteristics of each automaton (number of states and of silent transitions), the amount of precomputation time used by our approach, and the average time (over 400

	Example2	Example3	Example4	Example6	Example8
#State/#Silent Trans	3/6	4/6	4/7	7/10	7/5
Precomputation Time	173.25s	0.38s	791.06s	11.01s	4.96s
Actions `DiagOTA`	0.014s	0.019s	0.029s	0.17s	0.15s
Actions `TripakisDOTA`	0.020s	0.078s	0.049s	0.26s	0.042
Ratio (actions)	0.73	0.25	0.59	0.64	3.71
Delays `DiagOTA`	0.000012s	0.0000011s	0.000011s	0.000011s	0.000012s
Delays `TripakisDOTA`	0.032s	0.057s	0.049s	0.30s	0.033s
Ratio (delays)	0.0004	0.0002	0.0002	0.00003	0.0004

Fig. 3. Bench for 5 examples over 400 runs with 10 to 20 actions

random runs) used in the two approaches to simulate action- and delay transitions.

As could be expected, our approach outperforms the approach of [26] on delay transitions by several orders of magnitude in all cases. The performances of both approaches are comparable when simulating action transitions.

The precomputation phase of our approach is intrinsically very expensive. In our examples, it takes from less than a second to more than 13 minutes, and it remains to be understood which factors make this precomputation phase more or less difficult. We may also refine our implementation of the computation of Parikh images, which is heavily used in the precomputation phase.

6 Conclusion and future works

In this paper, we presented a novel approach to fault diagnosis for one-clock timed automata; it builds on a kind of powerset construction for automata over timed domains, using our new formalism of *timed sets* to represent the evolution of the set of reachable configurations of the automaton. Our prototype implementation shows the feasibility of our approach on small examples.

There remains space for improvements in many directions: first, our implementation can probably be made more efficient on the precomputation phase, and at least we need to better understand why some very small examples are so hard to handle.

A natural continuation of this work is an extension to n-clock timed automata. This is not immediate, as it requires a kind of *timed zone*, and an adaptation of our operator $\bowtie$. Another possible direction of research could target priced timed automata, with the aim of monitoring the cost of the execution in the worst case.

References

1. Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, April 1994.

2. Gerd Behrmann, Alexandre David, Kim Guldstrand Larsen, John Håkansson, Paul Pettersson, Wang Yi, and Martijn Hendriks. Uppaal 4.0. In *Proceedings of the 3rd International Conference on Quantitative Evaluation of Systems (QEST'06)*, pages 125–126. IEEE Comp. Soc. Press, September 2006.
3. Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In Jörg Desel, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Lectures on Concurrency and Petri Nets*, volume 2098 of *Lecture Notes in Computer Science*, pages 87–124. Springer-Verlag, 2004.
4. Albert Benveniste, Éric Fabre, Stefan Haar, and Claude Jard. Diagnosis of asynchronous discrete event systems: A net-unfolding approach. *IEEE Transactions on Automatic Control*, 48(5):714–727, 2003.
5. Nathalie Bertrand, Serge Haddad, and Engel Lefaucheux. Foundation of diagnosis and predictability in probabilistic systems. In Venkatesh Raman and S. P. Suresh, editors, *Proceedings of the 34th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'14)*, volume 29 of *Leibniz International Proceedings in Informatics*, pages 417–429. Leibniz-Zentrum für Informatik, December 2014.
6. Patricia Bouyer, Fabrice Chevalier, and Deepak D'Souza. Fault diagnosis using timed automata. In Vladimiro Sassone, editor, *Proceedings of the 8th International Conference on Foundations of Software Science and Computation Structure (FoSSaCS'05)*, volume 3441 of *Lecture Notes in Computer Science*, pages 219–233. Springer-Verlag, April 2005.
7. Patricia Bouyer, Samy Jaziri, and Nicolas Markey. On the determinization of timed systems. In Alessandro Abate and Gilles Geeraerts, editors, *Proceedings of the 15th International Conferences on Formal Modelling and Analysis of Timed Systems (FORMATS'17)*, volume 10419 of *Lecture Notes in Computer Science*, pages 25–41. Springer-Verlag, September 2017.
8. Franck Cassez. A note on fault diagnosis algorithms. In *Proceedings of the 48th IEEE Conference on Decision and Control (CDC'09)*, pages 6941–6946. IEEE Comp. Soc. Press, December 2009.
9. Yi-Liang Chen and Gregory Provan. Modeling and diagnosis of timed discrete event systems – a factory automation example. In *Proceedings of the 1997 American Control Conference (ACC'97)*, pages 31–36. IEEE Comp. Soc. Press, June 1997.
10. Edmund M. Clarke, E. Allen Emerson, and Joseph Sifakis. Model checking: algorithmic verification and debugging. *Communications of the ACM*, 52(11):74–84, November 2009.
11. Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model checking*. MIT Press, 2000.
12. Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem. *Handbook of Model Checking*. Springer-Verlag, April 2018.
13. Jean-Christophe Filliâtre. Deductive software verification. *International Journal on Software Tools for Technology Transfer*, 13(5):397–403, October 2011.
14. Olivier Finkel. Undecidable problems about timed automata. In Eugene Asarin and Patricia Bouyer, editors, *Proceedings of the 4th International Conferences on Formal Modelling and Analysis of Timed Systems (FORMATS'06)*, volume 4202 of *Lecture Notes in Computer Science*, pages 187–199. Springer-Verlag, September 2006.
15. Sahika Genc and Stéphane Lafortune. Predictability of event occurrences in partially-observed discrete-event systems. *Automatica*, 45(2):301–311, February 2009.

16. Thomas A. Henzinger and Joseph Sifakis. The embedded systems design challenge. In Jayadev Misra, Tobias Nipkow, and Emil Sekerinski, editors, *Proceedings of the 14th International Symposium on Formal Methods (FM'06)*, volume 4085 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, August 2006.
17. Charles Antony Richard Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, October 1969.
18. Stéphane Lafortune, Feng Lin, and Christoforos N. Hadjicostis. On the history of diagnosability and opacity in discrete event systems. *Annual Reviews in Control*, 45:257–266, 2018.
19. Martin Leucker and Christian Schallart. A brief account of runtime verification. *Journal of Logic and Algebraic Programming*, 78(5):293–303, May 2009.
20. Jan Lunze and Jochen Schröder. State observation and diagnosis of discrete-event systems described by stochastic automata. *Discrete Event Dynamic Systems*, 11(4):319–369, October 2001.
21. Sriram Narasimhan and Gautam Biswas. Model-based diagnosis of hybrid systems. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 37(3):348–361, May 2007.
22. Meera Sampath, Stéphane Lafortune, and Demosthenis Teneketzis. Active diagnosis of discrete-event systems. *IEEE Transactions on Automatic Control*, 43(7):908–929, July 1998.
23. Meera Sampath, Raja Sengupta, Stéphane Lafortune, Kasim Sinnamohideen, and Demosthenis Teneketzis. Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, September 1995.
24. Meera Sampath, Raja Sengupta, Stéphane Lafortune, Kasim Sinnamohideen, and Demosthenis Teneketzis. Failure diagnosis using discrete-event models. *IEEE Transactions on Computers*, 35(1):105–124, January 1996.
25. Jan Tretmans. Conformance testing with labelled transition systems: Implementation relations and test generation. *Computer Networks and ISDN Systems*, 29(1):49–79, 1996.
26. Stavros Tripakis. Description and schedulability analysis of the software architecture of an automated vehicle control system. In Alberto L. Sangiovani-Vincentelli and Joseph Sifakis, editors, *Proceedings of the 2nd International Conference on Embedded Software (EMSOFT'02)*, volume 2491 of *Lecture Notes in Computer Science*, pages 123–137. Springer-Verlag, October 2002.
27. Stavros Tripakis. Folk theorems on the determinization and minimization of timed automata. *Information Processing Letters*, 99(6):222–226, September 2006.
28. Janan Zaytoon and Stéphane Lafortune. Overview of fault diagnosis methods for discrete event systems. *Annual Reviews in Control*, 37(2):308–320, 2013.

A Appendix

A.1 Proof of Proposition 6

Proposition 6. *Let E and E' be regular unions of intervals, K be an interval, and $d \in \mathbb{Q}$. Then $E \cup E'$, $\overline{E}$, $E + d$ and $E - K$ are regular unions of intervals.*

Proof. Write $E = (I, J, p, q)$ and $E' = (I', J', p', q')$, so that

$$E = I \cup \bigcup_{k=q}^{+\infty} J - k \cdot p \qquad\qquad E' = I' \cup \bigcup_{k=q'}^{+\infty} J' - k' \cdot p'.$$

We can write $p = \frac{a}{b}$ and $p' = \frac{a'}{b'}$, with a, b, a' and b' in $\mathbb{N}$. Let

$$N = \min\{n \in \mathbb{N} \mid n \cdot b \cdot a' \geq q \text{ and } n \cdot b' \cdot a \geq q'\}.$$

Then we can write

$$E = \left(I \cup \bigcup_{k=q}^{N \cdot b \cdot a' - 1} J - k \cdot p \right) \cup \bigcup_{k=N}^{+\infty} \left[\left(\bigcup_{r=0}^{b \cdot a' - 1} J - r \cdot p \right) - k \cdot a \cdot a' \right].$$

and

$$E' = \left(I' \cup \bigcup_{k=q'}^{N \cdot b' \cdot a - 1} J' - k \cdot p' \right) \cup \bigcup_{k=N}^{+\infty} \left[\left(\bigcup_{r=0}^{b' \cdot a - 1} J' - r \cdot p' \right) - k \cdot a \cdot a' \right].$$

Since $J \subseteq (-p; 0]$, we have $\bigcup_{r=0}^{b \cdot a' - 1} J - r \cdot p \subseteq (-b \cdot a' \cdot p; 0] = (a \cdot a'; 0]$, and similarly for J'. Taking the union of the equalities above, we get an expression of $E \cup E'$ under the form $I'' \cup \bigcup_{k=N}^{+\infty} (J'' - k \cdot (a \cdot a')$, which proves that $E \cup E'$ is a regular union of intervals.

We now prove the result for $\overline{E}$: again writing $E = I \cup \bigcup_{k=q}^{+\infty} J - k \cdot p$, thanks to the constraints imposed on I and J, we have

$$\overline{E} = (\mathbb{1}(-q \cdot p) \setminus I) \cup \bigcup_{k=q}^{+\infty} ((-p; 0] \setminus J) - k \cdot p.$$

The proofs for $E + d$ and $E - K$ follow similar arguments. □

A.2 Proof of Lemma 9

Lemma 9. – *If $E \leq F$ (in particular, if $E \subseteq \mathbb{R}_{\leq 0}$ and $F \subseteq \mathbb{R}_{\geq 0}$), then $E \bowtie F = E - F$.*
– *if $E > F$, then $E \bowtie F = \emptyset$.*
– *If E and F are two intervals, then $E \bowtie F$ is an interval.*

– *IF E is a regular union of intervals and J is an interval, then $E \bowtie F$ is a regular union of intervals.*
– *If $F' \subseteq \mathbb{R}_{\geq 0}$, then $(E \bowtie F) \bowtie F' = (E \bowtie F) - F'$.*

Proof. The first three claims are trivial from the definition of $E \bowtie F$. The fourth claim follows from Prop. 6. Finally, the last claim is a consequence of the first one (because $E \bowtie F \subseteq \mathbb{R}_{\leq 0}$). □

A.3 Proof of Lemma 12

Lemma 12. *$\epsilon(M, w) \equiv M^w$ for all $w \in U^*$ and all linear timed marking M.*

Proof. For $d \in \mathbb{R}_{\geq 0}$ and $s \in S$, we have

$$M^w(d)(s) = \{v \in \mathbb{R}_{\geq 0} \mid \exists s' \in S.\, \exists d_o \leq d.\, \exists v' \in M(d_0)(s').\, (s', v') \xrightarrow{d-d_0}_w (s, v)\}.$$

Hence clearly $M^w(d)(s) \equiv (\emptyset, \uparrow 0)$ if $s \neq \mathsf{tgt}(w)$. When $s = \mathsf{tgt}(w)$, we have $M^w(d)(s) = \{v \in \mathbb{R}_{\geq 0} \mid \exists d_0 \leq d.\, \exists v' \in M(d_0)(\mathsf{src}(w)).\, (s', v') \xrightarrow{d-d_0}_w (s, v)\}$. By Lemma 10, this corresponds to $\epsilon(M, w)(s)(d)$. □

A.4 Proof of Lemma 14

Lemma 14. *Let $(E; \widehat{r})$ be a timed set with $E \subseteq \mathbb{R}_{\leq 0}$, and $w \in U^*$. Then*

$$\epsilon_1((E; \widehat{r}), w) = E - \sum_{J = (\widehat{g}, e) \in \mathbb{J}_{\widehat{r}}} \mathcal{J}_{\widehat{r}}(w)(J) \times \Phi(\widehat{g}, e) \subseteq \mathbb{R}_{\leq 0}.$$

Proof. The proof is by induction on w. The result is straightforward for $w = \bot$. Now, assume the result holds for some $w \in U^*$, and consider $w' = w \cdot e$.

First, if $e \in U_{id}$, then $\epsilon_1((E; \widehat{r}), w \cdot e) = \epsilon_1((E; \widehat{r}), w) \cap \underset{\sim}{e}$ (by definition of ϵ). Since $\epsilon_1((E; \widehat{r}), w) \subseteq \mathbb{R}_{\leq 0} \subseteq \underset{\sim}{e}$, we have $\epsilon_1((E; \widehat{r}), w \cdot e) = \epsilon_1((E; \widehat{r}), w)$. Since $\mathcal{J}_{\widehat{r}}(w \cdot e) = \mathcal{J}_{\widehat{r}}(w)$ when $e \in U_{id}$, our result follows.

For $e \in U_0$, we have

$$\epsilon_1((E; \widehat{r}), w \cdot e) = \epsilon_1((E; \widehat{r}), w) \bowtie (\epsilon_2(\widehat{r}, w)) \cap \widehat{e} \cap \underset{\sim}{e}).$$

Since $\epsilon_1((E; \widehat{r}), w) \subseteq \mathbb{R}_{\leq 0}$ and $\widehat{r} \subseteq \mathbb{R}_{\geq 0}$, Lemma 9 entails

$$\epsilon_1((E; \widehat{r}), w \cdot e) = \epsilon_1((E; \widehat{r}), w) - ((\epsilon_2(\widehat{r}, w)) \cap \widehat{e} \cap \underset{\sim}{e}).$$

This precisely corresponds to the effect of adding $\{(\epsilon_2(\widehat{r}, e), e)\}$ to the multiset $\mathcal{J}_{\widehat{r}}(w)$. □

A.5 Proof of Lemma 15

Lemma 15. *Let E be a regular union of intervals, $\widehat{r}$ and $\widehat{r}'$ be two elements of $\widehat{\mathbb{R}}_{\geq 0}$, and $\mathcal{L} \subseteq \mathcal{L}(\mathcal{A}_\epsilon)$ be a* regular *language. Then $\epsilon_1(M_{(s,E,\widehat{r})}, \mathcal{L}^{\widehat{r}'}_{\widehat{r}})$ is a regular union of intervals.*

Proof. The proof of this result is in two parts: we first express $\mathcal{L}^{\widehat{r}'}_{\widehat{r}}$ as the language of a finite automaton, and then—by a tedious proof—express $\epsilon_1(M_{(s,E,\widehat{r})}, \mathcal{L}^{\widehat{r}'}_{\widehat{r}})$ as a regular union of intervals.

Finite automaton for $\mathcal{L}^{\widehat{r}'}_{\widehat{r}}$. We assume that $\mathcal{L}$ is accepted by the automaton $\mathcal{B} = (S, \{s\}, U, \{s'\})$ obtained from $\mathcal{A}_\epsilon$ by imposing an initial state s and a single accepting state s'; this is enough for our purpose, and is easily generalized to any regular $\mathcal{L} \subseteq \mathcal{L}(\mathcal{A}_\epsilon)$. We decorate all states with elements of $\widehat{\mathbb{R}}_{\geq 0}$ in order to derive a finite automaton accepting $\mathcal{L}^{\widehat{r}'}_{\widehat{r}}$. More precisely, we consider the automaton $\mathcal{B}^{\widehat{r}'}_{\widehat{r}} = (S \times \mathbb{J}_{\widehat{r}}, (s, \widehat{r}), U', (s', \widehat{r}'))$ where

$$\begin{aligned} U' =& \{([q, \widehat{g}], \widehat{e}, \underbrace{e}, (\widehat{g}, e, \updownarrow 0), 0, [q', \updownarrow 0]) \mid e = (q, \widehat{e}, \underbrace{e}, 0, \epsilon, q') \in U \text{ and } \widehat{g} \cap \underbrace{e} \neq \emptyset\} \\ & \cup \{([q, \widehat{g}], \widehat{e}, \underbrace{e}, (\widehat{g}, e, \widehat{g}'), id, [q', \widehat{g}']) \mid e = (q, \widehat{e}, \underbrace{e}, id, \epsilon, q') \in U \text{ and } \widehat{g} \cap \underbrace{e} \neq \emptyset\}. \end{aligned}$$

This automaton accepts words in $(\widehat{\mathbb{R}}_{\geq 0} \times U \times \widehat{\mathbb{R}}_{\geq 0})^*$. Writing $\pi_U \colon \widehat{\mathbb{R}}_{\geq 0} \times U \times \widehat{\mathbb{R}}_{\geq 0} \to U$ for the projection on the second element of this alphabet (and extending it to sequences in the natural way), it is easily observed that $\pi_U(\mathcal{L}(\mathcal{B}^{\widehat{r}'}_{\widehat{r}})) = \mathcal{L}(\mathcal{B})^{\widehat{r}'}_{\widehat{r}}$.

Defining $\epsilon_1(M_{(s,E,\widehat{r})}, [\mathcal{L}(s, s')]^{\widehat{r}'}_{\widehat{r}})$ as a regular union of intervals. We now focus on $\epsilon_1(M_{(s,E,\widehat{r})}, [\mathcal{L}(s, s')]^{\widehat{r}'}_{\widehat{r}}) = \epsilon_1((E, \widehat{r}), [\mathcal{L}(s, s')]^{\widehat{r}'}_{\widehat{r}})$, which we write η for the sake of readability.

For any $\widehat{g} \in \mathbb{J}_{\widehat{r}} = \{\updownarrow 0, \widehat{r}\} \cup \widehat{G_{id}}$ and any $\underbrace{g'} \in \underbrace{G_{id}}$, we define

$$W^{id}_{\widehat{g}, \widehat{g}'} = \{w = (\widehat{g}_1, e_1, \widehat{g}'_1) \cdot (\widehat{g}_2, e_2, \widehat{g}'_2) \cdots (\widehat{g}_n, e_n, \widehat{g}'_n) \mid \widehat{g}_n = \widehat{g} \text{ and } \min_{1 \leq i \leq n} \underbrace{e_i} = \underbrace{g'}\}.$$

We decompose $[\mathcal{L}(s, s')]^{\widehat{r}'}_{\widehat{r}}$ as the union of

$$\bigcup_{\underbrace{g'} \in \underbrace{G_{id}}} [\mathcal{L}(s, s')]^{\widehat{r}'}_{\widehat{r}} \cap W^{id}_{\widehat{r}', \widehat{g}'}$$

(all runs that do not contain resetting transitions) and

$$\bigcup_{\underbrace{g'} \in \underbrace{G_{id}}} \bigcup_{(\widehat{g}, e) \in \mathbb{J}_{\widehat{r}} \times U_0} [\mathcal{L}(s, s')]^{\widehat{r}'}_{\widehat{r}} \cap \left[W^{id}_{\widehat{g}, \widehat{g}'} \times \{(\widehat{g}, e, \updownarrow 0)\} \times (\mathbb{J}_{\widehat{r}} \times U \times \mathbb{J}_{\widehat{r}})^* \right]$$

where the right-hand side of the intersection contains all paths containing at least one resetting transition labelled $(\widehat{g}, e, \updownarrow 0)$. Using this decomposition, we get

$$\begin{aligned} \eta = & \bigcup_{\underbrace{g'} \in \underbrace{G_{id}}} \bigcup_{w \in [\mathcal{L}(s,s')]^{\widehat{r}'}_{\widehat{r}} \cap W^{id}_{\widehat{r}', \widehat{g}'}} \epsilon_1((E, \widehat{r}), \pi_U(w)) \cup \\ & \bigcup_{\underbrace{g'} \in \underbrace{G_{id}}} \bigcup_{(\widehat{g}, e) \in \mathbb{J}_{\widehat{r}} \times U_0} \bigcup_{w \in [\mathcal{L}(s,s')]^{\widehat{r}'}_{\widehat{r}} \cap W^{id}_{\widehat{r}', \widehat{g}'} \times \{(\widehat{g}, e, \updownarrow 0)\} \times (\mathbb{J}_{\widehat{r}} \times U \times \mathbb{J}_{\widehat{r}})^*} \epsilon_1((E, \widehat{r}), \pi_U(w)) \end{aligned}$$

In the first part, $\pi_U(w)$ contains only non-resetting transitions, so that we have $\epsilon_1((E, \widehat{r}), \pi_U(w)) = E \cap \underbrace{g'}$. This is a finite union of regular intervals.

As for the second part, decomposing $w \in W^{id}_{\widehat{r}', \widehat{g}'} \times \{(\widehat{g}, e, \updownarrow 0)\} \times (\mathbb{J}_{\widehat{r}} \times U \times \mathbb{J}_{\widehat{r}})^*$ as $u \cdot (\widehat{g}, e, \updownarrow 0) \cdot v$, we have

$$\epsilon_1((E, \widehat{r}), \pi_U(w)) = \epsilon_1(\epsilon(\epsilon((E, \widehat{r}), \pi_U(u)), e), \pi_U(v)).$$

Since $\pi_U(u)$ contains non-resetting transitions, we have $\epsilon((E,\widehat{r}),\pi_U(u)) = E \cap \mathcal{g}' \subseteq \mathbb{R}_{\leq 0}$, and since e is a resetting transition, we get

$$\epsilon_1((E,\widehat{r}),\pi_U(w)) = \epsilon_1(((E \cap \mathcal{g}') \bowtie \Phi(\widehat{g},e),\uparrow\!0),\pi_U(v)).$$

Hence it remains to prove that for all $\mathcal{g}' \in \underbrace{G_{id}}$, the set

$$\eta' = \bigcup_{(\widehat{g},e)\in\mathbb{J}_{\widehat{r}}\times U_0} \epsilon_1\left(((E \cap \mathcal{g}') \bowtie \Phi(\widehat{g},e),\uparrow\!0),\pi_U(Q)\right),$$

where $Q = (W^{id}_{\widehat{g},\widehat{g}'} \times \{(\widehat{g},e,\uparrow\!0)\})\backslash[\mathcal{L}(s,s')]^{\widehat{r}'}_{\widehat{r}}$ is the left-quotient of $[\mathcal{L}(s,s')]^{\widehat{r}'}_{\widehat{r}}$ by $W^{id}_{\widehat{g},\widehat{g}'} \times \{(\widehat{g},e,\uparrow\!0)\}$, is a regular union of intervals.

Write $E_{e,\widehat{g},\mathcal{g}'} = (E \cap \mathcal{g}') \bowtie \Phi(\widehat{g},e) \subseteq \mathbb{R}_{\leq 0}$. From Lemma 14, we derive

$$\epsilon_1((E_{e,\widehat{g},\mathcal{g}'},\uparrow\!0),\pi_U(w)) = E_{e,\widehat{g},\mathcal{g}'} - \sum_{J\in\mathbb{J}_{\widehat{r}}\times U_0} \mathcal{J}_{\uparrow\!0}(\pi_U(w))(J) \times \Phi(J).$$

Hence

$$\begin{aligned}
\eta' &= \bigcup_{w\in Q} E_{e,\widehat{g},\mathcal{g}'} - \sum_{J\in\mathbb{J}_{\widehat{r}}\times U_0} \mathcal{J}_{\uparrow\!0}(\pi_U(w))(J) \times \Phi(J) \\
&= E_{e,\widehat{g},\mathcal{g}'} - \bigcup_{w\in Q} \sum_{J\in\mathbb{J}_{\widehat{r}}\times U_0} \mathcal{J}_{\uparrow\!0}(\pi_U(w))(J) \times \Phi(J) \\
&= E_{e,\widehat{g},\mathcal{g}'} - \bigcup_{P\in\mathbf{p}(Q)} \sum_{(\widehat{n},e)\in\mathbb{J}_{\widehat{r}}\times U_0} \sum_{\widehat{n}'\in\mathbb{J}_{\widehat{r}}} P(\widehat{n},e,\widehat{n}') \times \Phi(\widehat{n},e),
\end{aligned}$$

where $\mathbf{p}(Q)$ is the set of Parikh vectors of Q (i.e., for each $w \in Q$, $\mathbf{p}(w)$ is the multiset of states visited along w, and $\mathbf{p}(Q)$ is the set of all those multisets). The latter equality is obtained by observing that $\mathcal{J}_{\uparrow\!0}(\pi_U(w))(\widehat{n},e)$ is the number of occurrences of a state of the form $(\widehat{n},e,\widehat{n}')$ (for some $\widehat{n}'$) along w.

Now, the set $Q = (W^{id}_{\widehat{g},\widehat{g}'} \times \{(\widehat{g},e,\uparrow\!0)\})\backslash[\mathcal{L}(s,s')]^{\widehat{r}'}_{\widehat{r}}$ is regular, so that by Parikh's theorem, $\mathbf{p}(Q)$ is a semi-linear set; it can be written $\mathbf{p}(Q) = \bigcup_{i=1}^{c}(\mathbf{p}^i_0 + \sum_{j=1}^{d_i} \mathbf{p}^i_j \times \mathbb{N})$, where $\mathbf{p}^i_j \in \mathbb{N}^{\mathbb{J}_{\widehat{r}}\times U_0\times\mathbb{J}_{\widehat{r}}}$ for all $1 \leq i \leq c$ and $0 \leq j \leq d_i$. Then:

$$\begin{aligned}
\eta' = E_{e,\widehat{g},\mathcal{g}'} - \bigcup_{i=1}^{c}\Big(&\sum_{(\widehat{n},e)\in\mathbb{J}_{\widehat{r}}\times U_0} \sum_{\widehat{n}'\in\mathbb{J}_{\widehat{r}}} \mathbf{p}^i_0(\widehat{n},e,\widehat{n}') \times \Phi(\widehat{n},e) + \\
&\bigcup_{(n_j)_{1\leq j\leq d_i}\in\mathbb{N}^{d_i}} \sum_{k=1}^{d_i} \sum_{(\widehat{n},e)\in\mathbb{J}_{\widehat{r}}\times U_0} \sum_{\widehat{n}'\in\mathbb{J}_{\widehat{r}}} n_k \cdot \mathbf{p}^i_j(\widehat{n},e,\widehat{n}') \times \Phi(\widehat{n},e)\Big)
\end{aligned}$$

We write $\Gamma_{e,\widehat{g},\mathcal{g}'}$ for the second term, so that $\eta' = E_{e,\widehat{g},\mathcal{g}'} - \Gamma_{e,\widehat{g},\mathcal{g}'}$. For $1 \leq i \leq c$ and $0 \leq j \leq d_i$, we let

$$K^i_j = \sum_{(\widehat{n},e)\in\mathbb{J}_{\widehat{r}}\times U_0} \sum_{\widehat{n}'\in\mathbb{J}_{\widehat{r}}} \mathbf{p}^i_j(\widehat{n},e,\widehat{n}') \times \Phi(\widehat{n},e).$$

We now consider two cases:

– first assume that for some i_0 and j_0 and some $(\widehat{n}_0, e_0) \in \mathbb{J}_{\widehat{\tau}} \times U_0$, it holds $\mathbf{p}^{i_0}_{j_0}(\widehat{n}_0, e_0, \uparrow 0) > 0$ and $\Phi(\widehat{n}_0, e_0)$ has positive length. Let $L_n = K^{i_0}_0 + n \cdot K^{i_0}_{j_0}$ for any $n \in \mathbb{N}$. Then clearly $L_n \subseteq \Gamma_{e,\widehat{g},g'}$ for all n. Moreover, by definition of i_0 and j_0, the length of $K^{i_0}_{j_0}$ is positive, so that the length of $n \cdot K^{i_0}_{j_0}$ tends to infinity. It follows that for some $\alpha \in \mathbb{R}_{\geq 0}$, $\uparrow\alpha \subseteq \bigcup_{n\in\mathbb{N}} L_n \subseteq \Gamma_{e,\widehat{g},g'}$.
Then $\Gamma_{e,\widehat{g},g'} \cap \downarrow\alpha \subseteq [0;\alpha]$, and it has finite granularity, so that it is a finite union of intervals.
– We now assume that for all i and j and all $(\widehat{n}, e) \in \mathbb{J}_{\widehat{\tau}} \times U_0$, either $\mathbf{p}^i_j(\widehat{n}, e, \uparrow 0) = 0$, or $\Phi(\widehat{n}, e)$ has length 0. All Then for all $1 \leq i \leq n$ and $1 \leq j \leq d_i$, K^i_j is a finite union of punctual intervals, so that η' is a regular union of intervals.

□

B Examples definition

B.1 Example2

```
# States. Size : 3
q0;q1;q2
# Transitions
q0;[0,2];0;q0;a
q0;[1,1];0;q1;a
q1;[1,2];0;q1;a
q1;[2,2];0;q2;a
q2;[0,inf[;0;q0;a
q0;]0,2];0;q0;b
q0;[0,inf[;0;q1;b
q1;]1,2];0;q0;b
q1;[0,2[;0;q1;b
q2;[0,0];0;q1;b
q2;]0,1];0;q2;b
q0;]2,inf[;0;q0;e
q0;[1,2[;0;q1;e
q0;[0,2[;1;q2;e
q1;]0,2];0;q0;e
q1;]2,inf[;0;q1;e
q2;[1,1];0;q0;e
```

B.2 Example3

```
# States. Size : 4
q0;q1;q2;q3
# Transitions
q0;]1,2];0;q1;a
q0;[2,2];1;q3;a
q1;[0,1];0;q1;a
```

```
q2;]1,2[;1;q1;a
q2;]0,2];0;q2;a
q2;[1,2];0;q3;a
q3;[1,1];1;q2;a
q0;[2,inf[;0;q0;b
q0;[2,2];1;q2;b
q2;[0,0];0;q3;b
q3;]1,2];1;q0;b
q3;]0,inf[;1;q1;b
q3;[0,0];1;q3;b
q0;[0,inf[;1;q0;e
q0;[2,inf[;1;q1;e
q0;[0,1[;1;q3;e
q1;]0,2];1;q1;e
q1;]0,inf[;1;q2;e
q2;[1,inf[;1;q0;e
```

B.3 Example4

```
# States. Size : 4
q0;q1;q2;q3
# Transitions
q0;[0,2[;0;q0;a
q0;]0,2[;1;q1;a
q0;]1,inf[;1;q2;a
q0;]1,2[;0;q3;a
q1;]1,2];0;q0;a
q1;[0,1];1;q1;a
q1;[0,2[;0;q2;a
q1;]0,1[;0;q3;a
q2;[1,inf[;1;q0;a
q2;[1,2[;0;q3;a
q3;[2,inf[;0;q0;a
q3;[1,2];0;q2;a
q0;]1,2];1;q1;b
q1;[0,1[;1;q2;b
q1;[0,2];0;q3;b
q2;[0,0];1;q0;b
q2;[0,0];0;q2;b
q3;[0,2[;1;q0;b
q3;]1,2];1;q2;b
q3;[0,2];1;q3;b
q0;]0,inf[;1;q0;e
q0;]0,2];1;q1;e
q0;[1,1];0;q2;e
q0;]2,inf[;0;q3;e
```

```
q1;[0,inf[;1;q0;e
q1;[0,2[;0;q1;e
q2;[0,inf[;0;q0;e
```

B.4 Example6

```
# States. Size : 7
q0;q1;q2;q3;q4;q5;q6
# Transitions
q0;]0,1[;1;q0;a
q0;]0,2];1;q4;a
q0;[2,2];1;q6;a
q1;[0,0];1;q0;a
q1;[1,2];0;q4;a
q1;[0,0];0;q5;a
q2;[1,2];1;q1;a
q2;[0,3[;0;q2;a
q2;]0,2[;0;q5;a
q3;[0,inf[;1;q0;a
q3;]0,3];1;q4;a
q4;[3,3];1;q1;a
q4;]1,2];0;q3;a
q4;[0,2[;0;q4;a
q4;[2,3];0;q6;a
q5;]0,2[;0;q0;a
q5;]1,3[;0;q3;a
q5;[0,3[;1;q6;a
q6;]1,2[;1;q0;a
q6;[3,inf[;0;q1;a
q6;[1,2];1;q2;a
q6;]0,3[;0;q3;a
q6;[3,3];1;q5;a
q0;]2,inf[;1;q3;b
q0;]0,2[;0;q4;b
q0;[0,2];1;q5;b
q1;[1,3];0;q0;b
q1;[3,inf[;0;q2;b
q1;[0,1];1;q6;b
q2;[0,1[;0;q0;b
q2;[2,3[;0;q1;b
q3;[0,2];0;q3;b
q3;]0,2[;1;q4;b
q3;]2,3[;1;q6;b
q4;[3,inf[;1;q2;b
q4;[0,3];1;q6;b
q5;[0,0];0;q0;b
```

```
q5;[1,1];1;q2;b
q5;]1,3[;1;q4;b
q5;[3,3];1;q6;b
q6;[1,3];1;q1;b
q6;]1,2];1;q3;b
q0;[1,inf[;1;q0;e
q0;[0,0];1;q2;e
q0;[1,2];0;q4;e
q1;]0,3];1;q0;e
q1;]2,3[;0;q1;e
```

B.5 Example8

```
# States. Size : 7
q0;q1;q2;q3;q4;q5;q6
# Transitions
q0;[2,2];0;q0;a
q0;[0,3[;1;q1;a
q0;[2,2];1;q3;a
q0;[1,3];1;q5;a
q1;]0,3];1;q0;a
q1;]0,2[;0;q5;a
q1;]2,3];1;q6;a
q2;]3,inf[;1;q1;a
q2;[2,2];1;q2;a
q2;]0,3[;1;q4;a
q2;]1,3[;1;q5;a
q3;]1,3[;0;q1;a
q3;]1,2[;0;q3;a
q4;[1,3];0;q0;a
q4;[0,3];0;q2;a
q4;[0,3[;1;q3;a
q4;]2,3];0;q4;a
q4;[0,3[;0;q5;a
q4;[2,inf[;0;q6;a
q5;[2,inf[;1;q0;a
q5;[0,0];1;q1;a
q5;]1,2[;1;q2;a
q5;]0,1];1;q5;a
q6;]1,2[;1;q1;a
q6;]1,2[;1;q3;a
q6;[0,1];0;q5;a
q6;]2,3[;0;q6;a
q0;[3,3];1;q2;b
q0;]2,3[;0;q4;b
q0;]2,3[;0;q5;b
```

```
q0;]2,3];1;q6;b
q1;]1,inf[;1;q4;b
q1;]0,2];0;q6;b
q2;[1,inf[;1;q0;b
q2;]3,inf[;0;q1;b
q2;[1,3[;1;q4;b
q2;[0,3[;0;q5;b
q3;[1,3[;1;q0;b
q3;]3,inf[;1;q2;b
q3;[2,inf[;1;q4;b
q4;]0,2[;1;q1;b
q4;]0,inf[;1;q2;b
q4;]0,2[;1;q3;b
q4;[1,3[;1;q4;b
q4;]3,inf[;0;q5;b
q5;[0,2];1;q1;b
q6;]1,3];1;q3;b
q6;[1,2[;1;q4;b
q6;]1,2];0;q5;b
q6;[0,1];1;q6;b
q0;[0,3[;1;q1;e
q0;]1,3];1;q3;e
q0;]0,3[;0;q5;e
q0;]3,inf[;1;q6;e
q1;]1,2[;0;q0;e
q1;]2,3];0;q2;e
q1;[1,2[;1;q3;e
q1;[0,1[;0;q5;e
q1;]0,3];1;q6;e
q2;[1,inf[;1;q0;e
```